Performance des étudiants

Corre, Feteira, Lenoir

Table des matières

1 Introduction

Nous allons dans ce rapport essayer de trouver le meilleur modèle de prédiction associé à notre base de données concernant la performance des étudiants au Portugal. L'objectif principal est donc : en ayant comme données les caractéristiques d'un étudiant de pouvoir lui prédire si il a plus de chances d'avoir une bonne ou une mauvaise note. En effet, pour cette étude, notre variable à prédire est la note moyenne de l'étudiant. Soit il aura une bonne note (supérieure à la moyenne de 10), soit une mauvaise note (inférieure à la moyenne). Tout d'abord, nous essayerons les modèles d'analyse factorielle discriminante tel que LDA, QDA, MDA ainsi qu'un modèle de régression logistique. Nous comparerons ensuite les différents modèles pour choisir le meilleur.

Ensuite, nous étudierons les differentes techniques d'arbres de classification supervisée tels que les arbres de décisions, les random forest et le boosting. Nous allons également les comparer afin de garder le meilleur modèle final.

Pour nos modèles en analyses factorielles discriminantes, nous les testerons à la fois sur les données avec toutes les variables mais également sur une base de données contenant seulement les variables qui semblaient le plus importantes pendant notre ACM du 1er semestre. Les variables retenues seront :

- `Medu`, `Fedu`, `Mjob`, `Fjob`
- `studytime`, `failures`, `paid`, `higher`, `absences`
- `freetime`, `goout`
- `moyenne_facteur`

2 Nettoyage et création de la variable à prédire

Initialement, on découpe la variable moyenne en 2 catégories : les notes inférieures à 10 et les notes supérieures à 10. De plus on crée les dataframes avec toutes les variables et avec les variables ressorties en ACM.

Table 1 – Effectif par modalité sur les notes

	< 10	**> 10**
Effectifs	345	664

3 Application des méthodes d'analyse factorielle discriminante

3.1 LDA

3.1.1 LDA base totale

Pour commencer, nous allons réaliser une LDA sur nos données complètes. Voici la matrice de confusion :

Table 2 – Matrice de confusion LDA

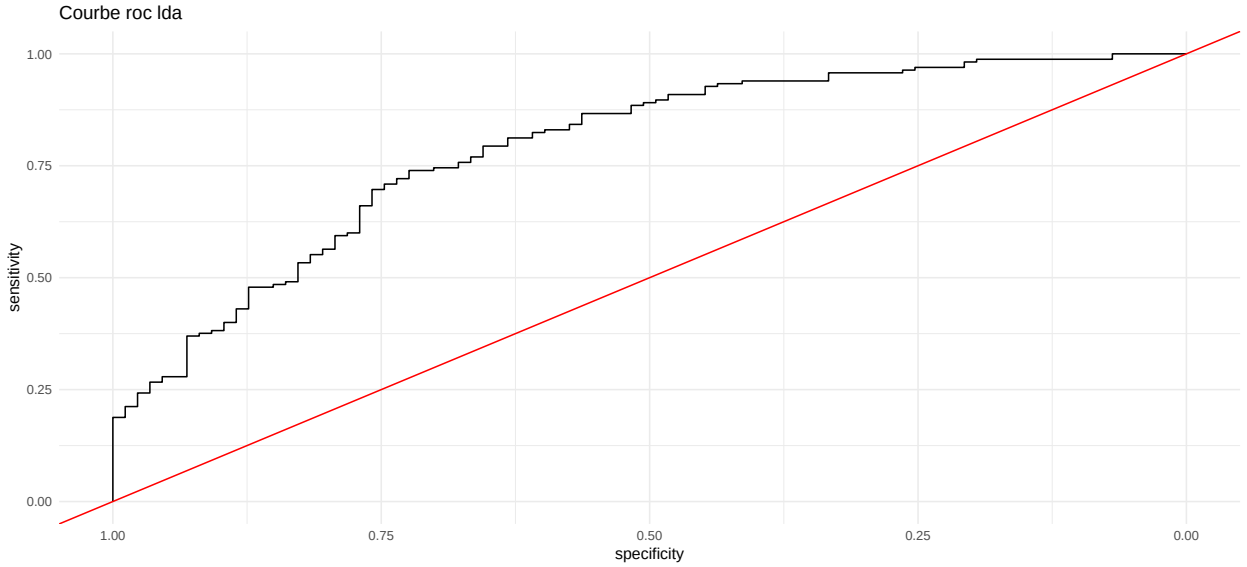
	Prediction	
	< 10	**> 10**
< 10	49	38
> 10	24	141

Table 3 – Matrice de confusion (proportions)

	Prediction	
	< 10	**> 10**
< 10	0.563	0.437
> 10	0.145	0.855

On remarque que pour ce modèle, nous prédisons bien les notes au dessus de la moyenne, toutefois les mauvaises notes sont un peu moins biens prédites.

— Le nombre de bonne prédiction est de 190, soit un taux de bonne affectation de 75.397%.
— 56.322% des personnes ayant une moyenne inférieure à 10 sont biens affectées. Ici, on parle de sensibilité.
— 85.455% des personnes ayant une moyenne supérieure à 10 sont bien affectées. Ici, on parle de spécificité.



Le modèle n'est pas mauvais mais pas non plus super efficace, surtout pour les personnes ayant une note sous la moyenne. L'aire sous la courbe ROC est de 0.79.

3.1.2 LDA variables intéressantes

Table 4 – Matrice de confusion LDA court

	Prediction	
	< 10	**> 10**
< 10	44	43
> 10	21	144

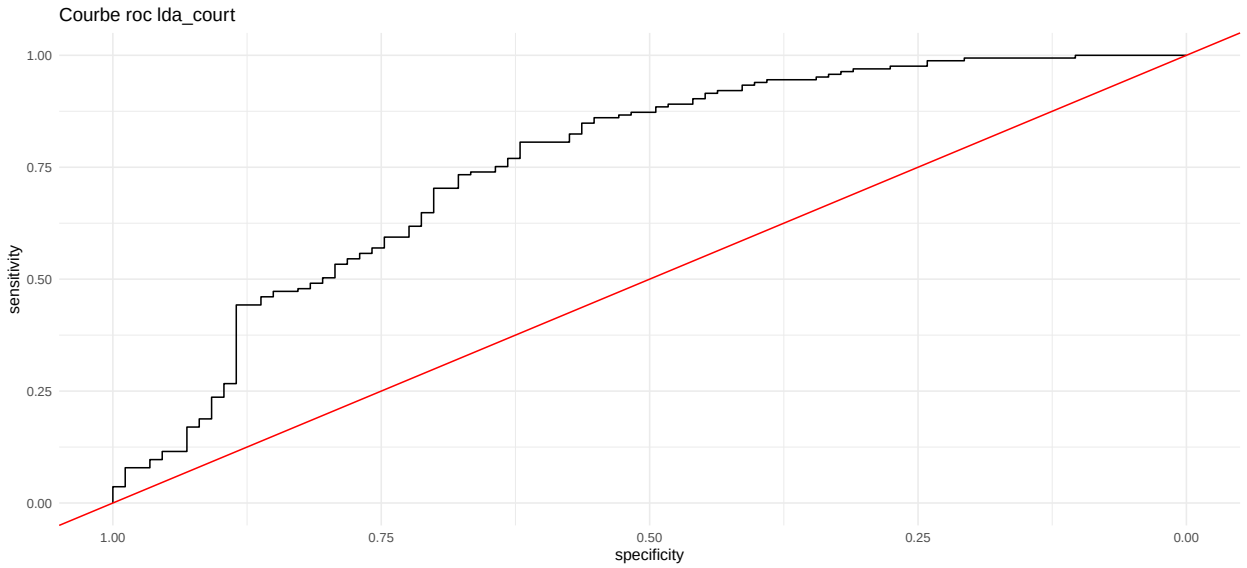
Table 5 – Matrice de confusion (proportions)

	Prediction	
	< 10	**> 10**
< 10	0.506	0.494
> 10	0.127	0.873

Ici, les bonnes notes sont biens prédites mais seulement la moitié des mauvaises notes sont bien prédites.

— Le nombre de bonnes prédictions est de 188, soit un taux de bonne affectation de 74.603%.
— La sensibilité est de 50.575% , c'est à dire le taux de personnes ayant une moyenne inférieure à 10 qui sont biens affectées.
— La spécificité est de 87.273% , c'est à dire le taux de personnes ayant une moyenne supérieure à 10 qui sont biens affectées.

Nous retrouvons toujours ce problème où notre prédiction sur les moyennes faibles est mauvaise.



Ici, l'aire sous la courbe observée est de 0.756. Ce qui est un tout petit peu moins que pour la LDA sur données complètes.

3.2 QDA

3.2.1 Base totale

TABLE 6 – Matrice de confusion QDA

	Prediction	
	< 10	**> 10**
< 10	51	36
> 10	37	128

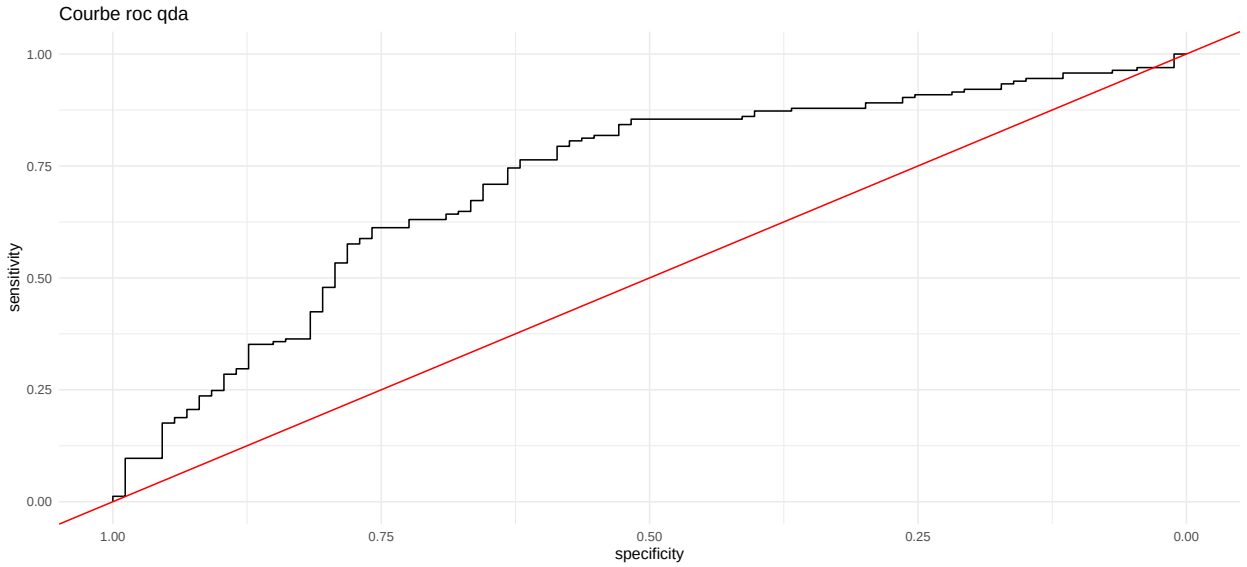
TABLE 7 – Matrice de confusion (proportions)

	Prediction	
	< 10	**> 10**
< 10	0.586	0.414
> 10	0.224	0.776

Nous remarquons que l'on prédit un peu mieux les mauvaises moyennes et un peu moins bien les bonnes moyennes. Au final, on a un moins bon modèle avec une erreur globale plus élevée.

— Le taux de bonnes prédictions est de 179, soit un taux de bonne affectation de 71.032%.

— 58.621% des personnes ayant une moyenne inférieure à 10 sont biens affectées.
— 77.576% des personnes ayant une moyenne supérieure à 10 sont biens affectées.



Comme vu dans le tableau, le modèle est un peu moins performant, avec une aire sous la courbe ROC égale à 0.715.

3.2.2 Variables ressorties en ACM

TABLE 8 – Matrice de confusion QDA variables ressorties en ACM

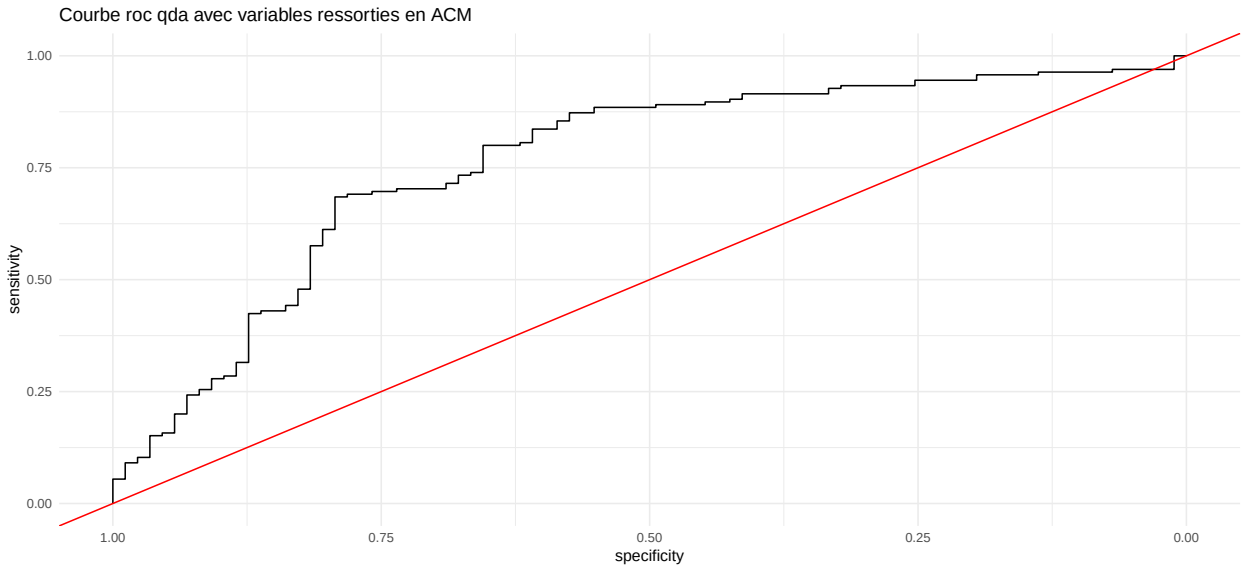
	Prediction	
	< 10	**> 10**
< 10	53	34
> 10	30	135

TABLE 9 – Matrice de confusion (proportions)

	Prediction	
	< 10	**> 10**
< 10	0.609	0.391
> 10	0.182	0.818

Nous remarquons que l'on prédit encore un peu mieux les mauvaises moyennes et un peu moins bien les bonnes moyennes. Au final, on a un bon modèle avec seulement 0.254% d'erreur globale.

— Le nombre de bonnes prédictions est de 188, soit un taux de bonne affectation de 74.603 %.
— 60.92 % des personnes ayant une moyenne inférieure à 10 sont biens affectées.
— 81.818 % des personnes ayant une moyenne supérieure à 10 sont biens affectées.



Nous avons un bon modèle avec une AUC de 0.761.

3.3 Comparaison de nos modèles:

Nous allons maintenant comparer tous nos modèles en utilisant la courbe ROC pour chacun de ceux-ci :

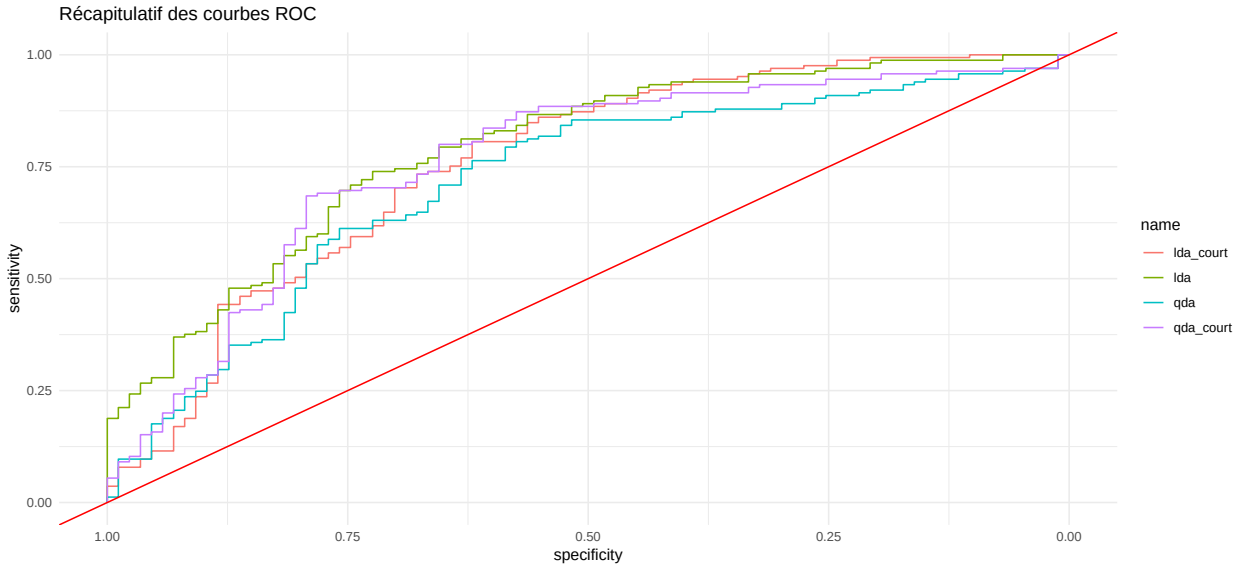


TABLE 10 – Comparaison des modèles AFD

	Erreur globale	AUC
LDA	0.246	0.790
LDA court	0.254	0.756
QDA	0.290	0.715
QDA court	0.254	0.761

Notre meilleur modèle est donc le tout premier modèle réalisé, c'est-à-dire le modèle LDA sur notre base de données complète avec une erreur de 0.246. Ensuite viennent les modèles LDA court et QDA court qui sont

quasiment similaires. Enfin, le moins bon modèle est le modèle QDA sur base complète avec une erreur de 0.29.

3.4 Modèle Logit

On a également décidé de tester nos données sur le modèle logit. Nous n'avons pas décidé de reprendre la base de données avec les coordonnées des individus de l'ACM car à la fin de ce modèle, nous allons faire une première interprétation de nos variables.

TABLE 11 – Matrice de confusion logit

	< 10	> 10
< 10	48	39
> 10	21	144

TABLE 12 – Proportions par ligne de la matrice de confusion logit

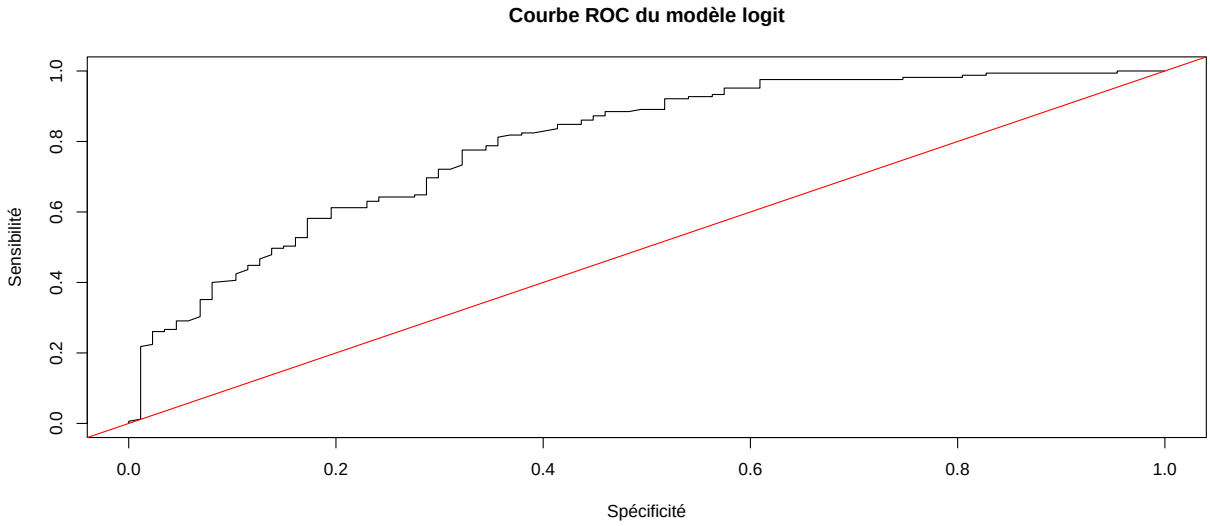
	< 10	> 10
< 10	0.552	0.448
> 10	0.127	0.873

On peut remarquer que les résultats sont similaires aux modèles LDA et QDA, en effet on prédit bien les bonnes notes mais moins bien les mauvaises.

```
##
## ================================================
##                          Dependent variable:
##                      ---------------------------
##                             moyenne_facteur
## ------------------------------------------------
## schoolMS                      -0.789***
##                                (0.214)
##
## Mjobhealth                     0.718*
##                                (0.417)
##
## Mjobother                      0.144
##                                (0.247)
##
## Mjobservices                  0.954***
##                                (0.293)
##
## Mjobteacher                    0.121
##                                (0.312)
##
## guardianmother                -0.569**
##                                (0.223)
##
## guardianother                  0.010
##                                (0.452)
##
## traveltimetrav > 1h            -1.138*
```

```
##                                (0.625)
##
## traveltimetrav 15-30            0.174
##                                (0.208)
##
## traveltimetrav 30-1h            0.282
##                                (0.341)
##
## studytimestud > 10h             0.313
##                                (0.407)
##
## studytimestud 2-5h              0.013
##                                (0.205)
##
## studytimestud 5-10h           0.905***
##                                (0.315)
##
## failures1                     -1.908***
##                                (0.302)
##
## failures2 ou 3                -2.758***
##                                (0.705)
##
## schoolsupyes                  -1.110***
##                                (0.273)
##
## higheryes                     1.010***
##                                (0.361)
##
## absencesabs ]5,10]             -0.394*
##                                (0.227)
##
## absencesabs ]10,75]            -0.088
##                                (0.294)
##
## Constant                        0.345
##                                (0.452)
##
## ------------------------------------------------
## Observations                     757
## Log Likelihood                -401.025
## Akaike Inf. Crit.              842.049
## ================================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

Lorsque l'on passe notre modèle sur toutes les données, on remarque que les variables school, failures (le fait de redoubler au moins une fois ici), higher (le fait de vouloir faire des études supérieures), studytime pour la modalité "5-10h" en comparaison à la modalité "< 2h" et Mjob pour la modalité "services" en comparaison à la modalité "at_home" semblent se différencier. Ces variables sont significatives. Nous retrouvons donc des résultats plutôt similaires à ce que l'on avait dans l'ACM du premier semestre. L'erreur du modèle logit est de 23.8%, ce qui est mieux que notre LDA. Pour conclure, voici une représentation de la courbe ROC de notre modèle logit.



4 Arbres de classification supervisée.

Danc cette partie, nous allons essayer de chercher nos meilleurs modèles en fonction des différentes familles. Nous commencerons par les arbres de décision en utilisant `Rpart`, puis les `random forest` et `bagging`, nous finirons par le `boosting`. Pour chaque partie, nous modifierons les paramètres afin de trouver le meilleur modèle.

En dernier lieu, nous comparerons tous nos meilleurs modèles de chaque famille afin de pouvoir choisir notre meilleur modèle final et l'appliquer à nos données.

4.1 Arbres de décisions

Après avoir fait un premier arbre, nous déterminons la valeur optimale de complexité de l'arbre pour en construire un nouveau.

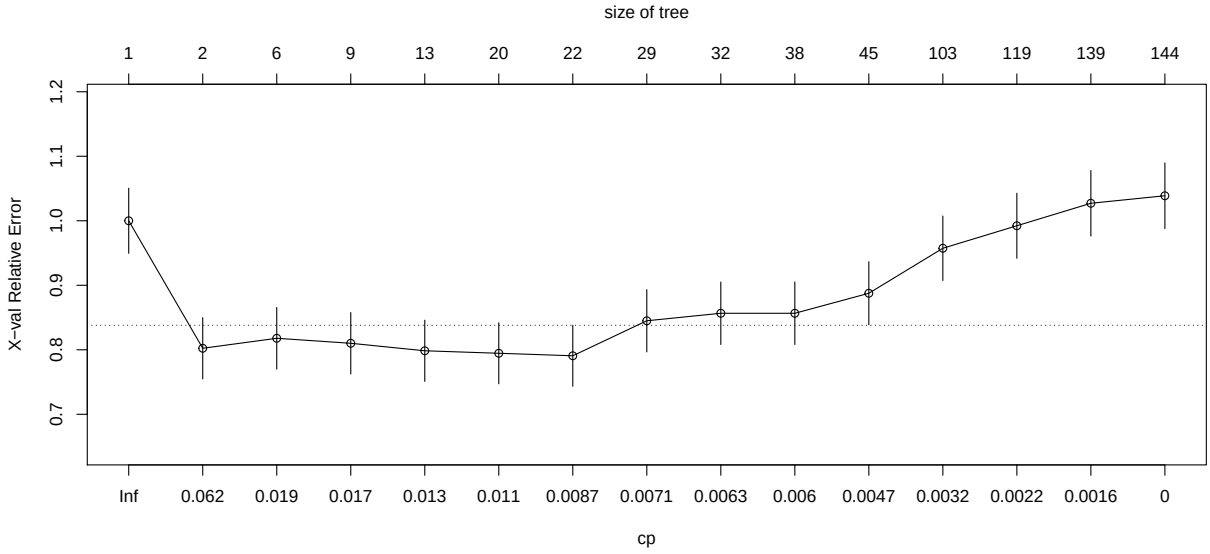
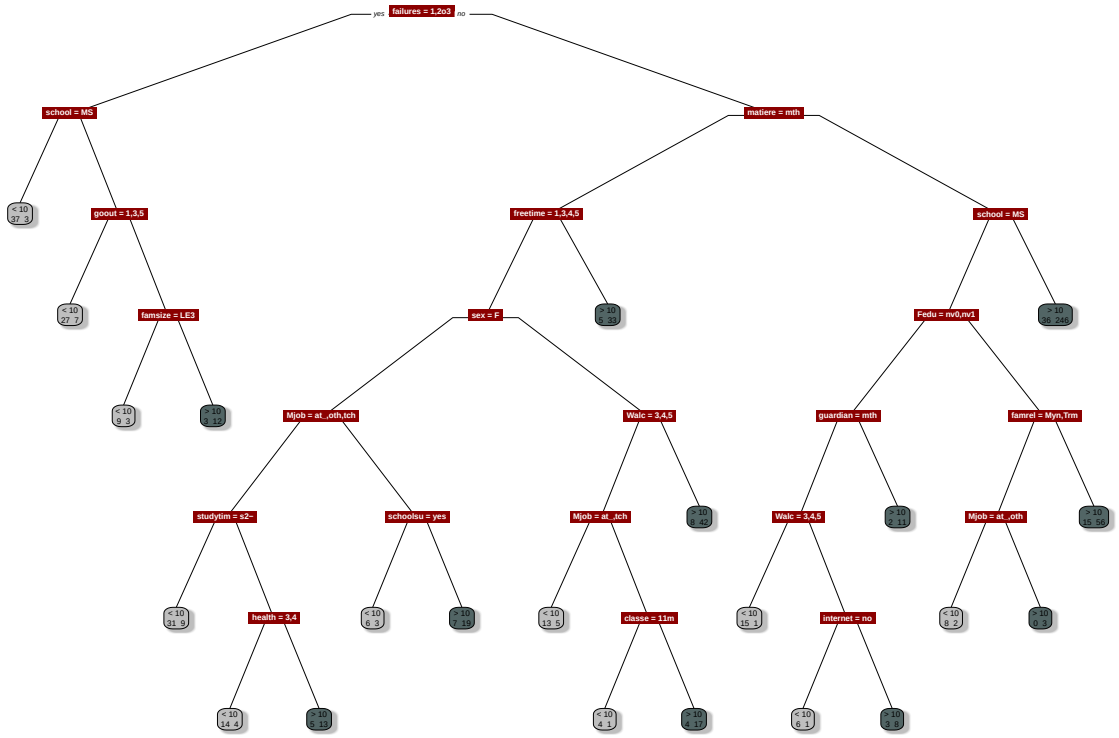


Table 13 – Complexité

CP	nsplit	rel error	xerror	xstd
0.198	0	1.000	1.000	0.051
0.019	1	0.802	0.802	0.048
0.018	5	0.709	0.818	0.048
0.016	8	0.655	0.810	0.048
0.012	12	0.593	0.798	0.047
0.010	19	0.512	0.795	0.047

Avec l'aide de ce graphique, le nombre de noeuds optimal pour élaguer notre arbre est de 22, pour une complexité de 0.0087.

Voici un aperçu de notre arbre final :



Nous allons maintenant tester notre arbre final sur les données tests pour déterminer s'il classe bien la moyenne de nos individus en fonction de leurs caractéristiques.

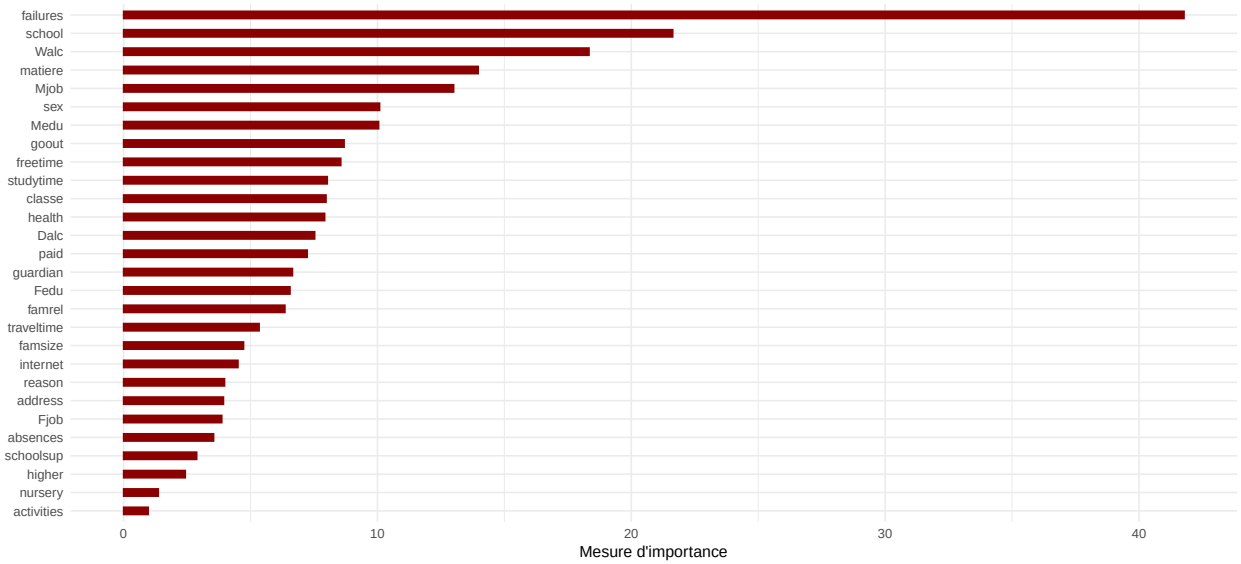
TABLE 14 – Matrice de confusion arbre de décision

	Réalité	
	< 10	> 10
< 10	42	30
> 10	45	135

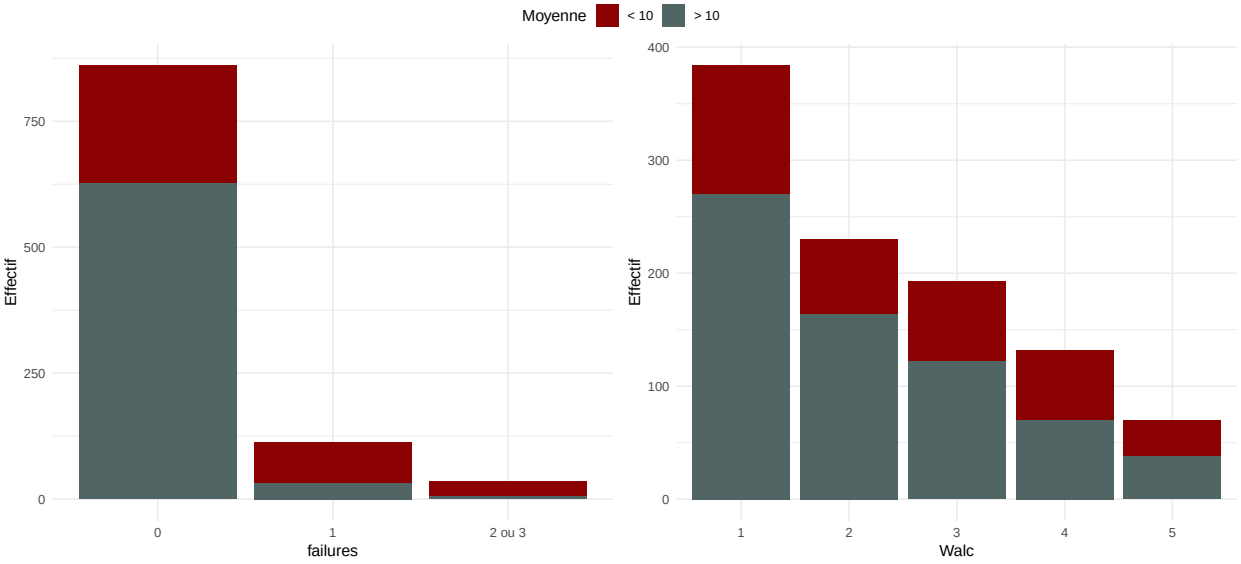
D'après la matrice de confusion sur nos données test, on remarque que :

- 70.238% de nos données sont biens prédites.
- La sensibilité est de 48.276%.
- La spécificité est de 81.818%.
- Nous avons une erreur moyenne de 29.762 % pour ce modèle.

4.1.0.1 Importance des variables



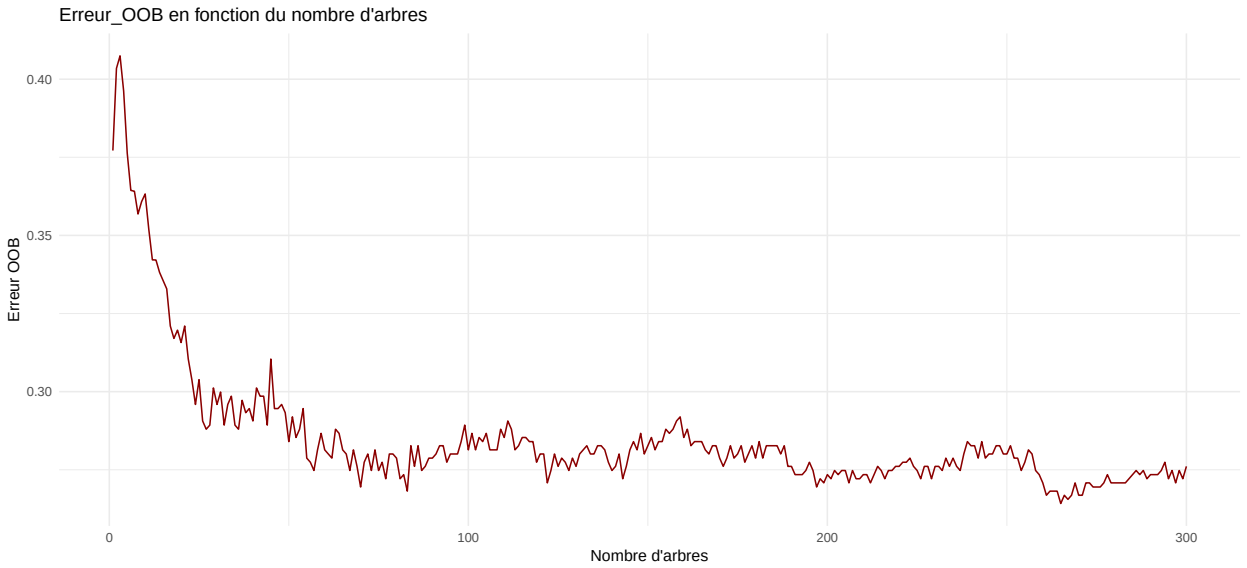
On remarque qu'au niveau de mesure d'importance, c'est la variable `failures` qui est la plus importante, suivie de `Walc` (consommation d'alcool le week-end) et `matière`.



Grâce à ces graphiques, on peut bien remarquer que pour la variable `failures` plus le nombre de redoublement est important, plus la proportion d'étudiants ayant une mauvaise note est importante. Pour 2 ou 3 redoublements, quasiment 100% des étudiants ont une moyenne en dessous de 10. En ce qui concerne la variable `Walc`, plus la consommation est importante, plus la part de mauvaises notes augmente également.

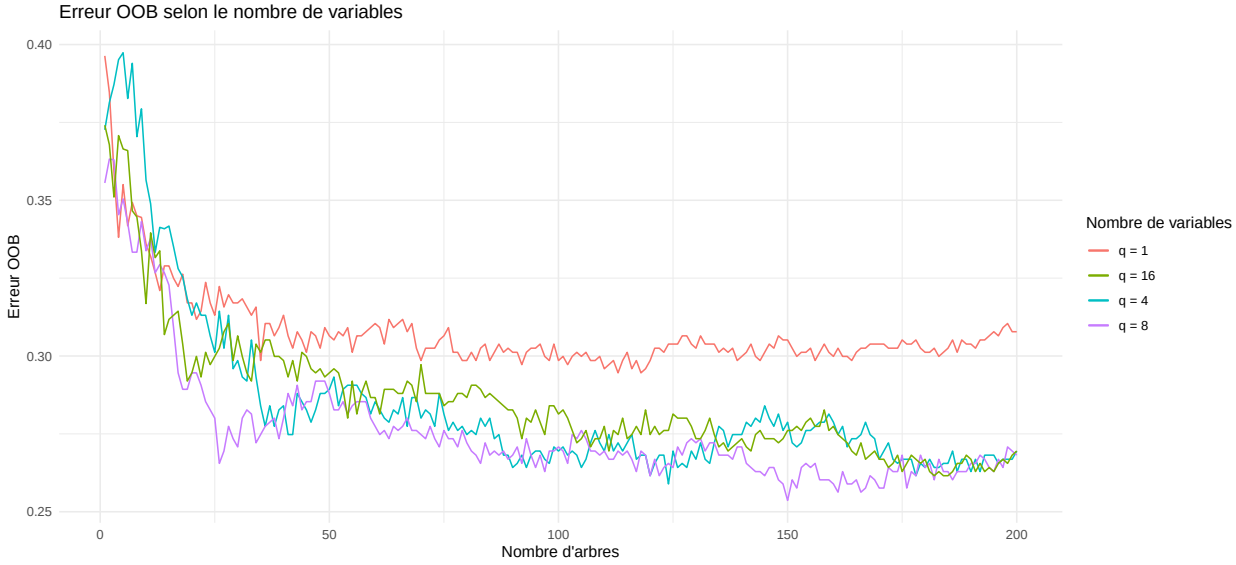
4.2 Fôrets aléatoires

On va tout d'abord choisir le nombre d'arbres qui semble optimal grâce à un graphique représentant l'erreur OOB en fonction du nombre d'arbres. Il semblerait que 200 arbres soit une bonne valeur.



Le meilleur nombre d'arbres semblerait être à environ 200. On a pas décidé de prendre moins car nous avons peu de données donc ce n'est pas très coûteux pour l'ordinateur de faire environ 50 arbres de plus.

Après avoir déterminer quel était le meilleur nombre d'arbres, nous determinons le nombre de variables à faire varier à chaque scission et on visualise les erreurs OOB. Il semblerait que prendre `mtry` $= 8$ soit la meilleure solution.



Un nombre de variables $= 8$ est le plus optimal pour un nombre d'arbre maximum de 200.

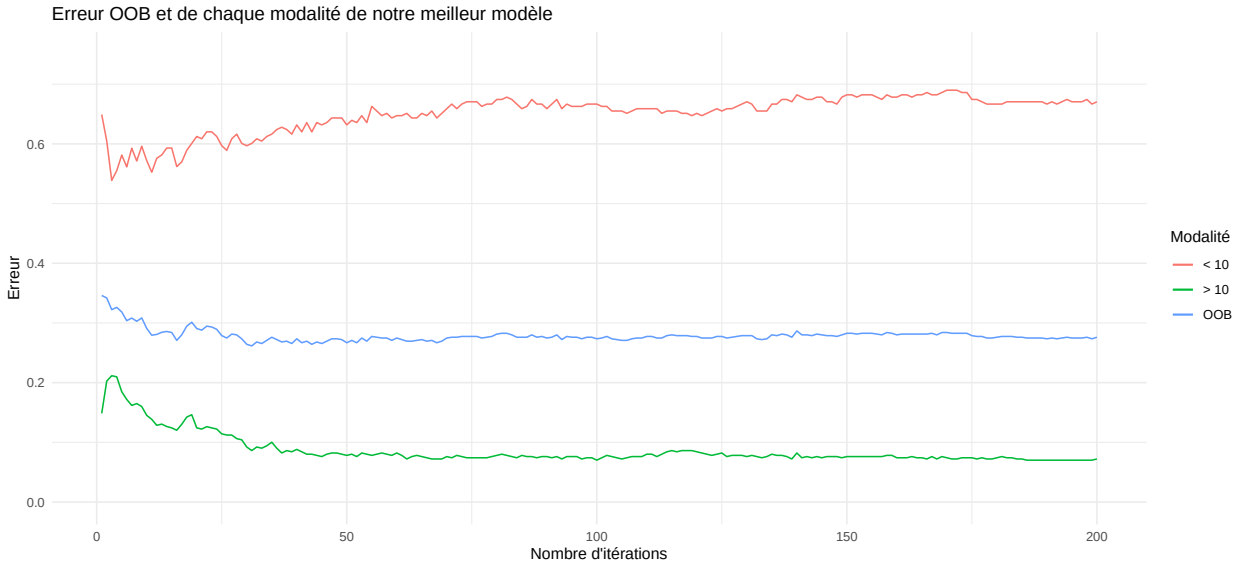
En ce qui concerne la profondeur de notre arbre, les résultats changeaient trop et il était impossible de choisir la bonne valeur, nous garderons donc une valeur qui était toujours faible, c'est à dire une profondeur de 50.

Voici un tableau récapitulatif des différents modèles de random forest afin de choisir le meilleur modèle :

TABLE 15 – Modèles random forest

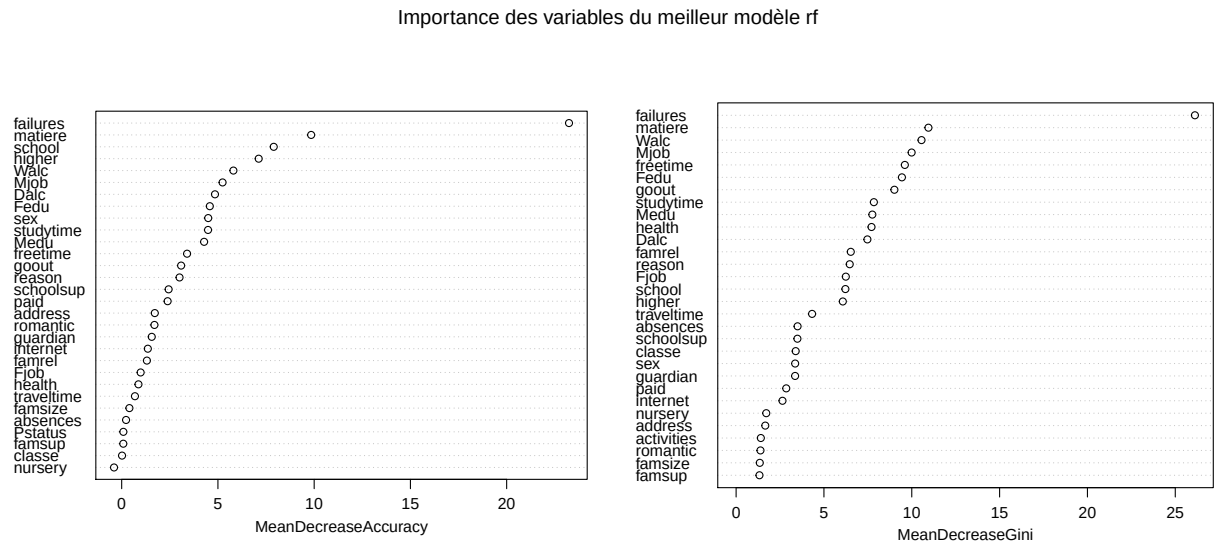
	Err_test	Err_OOB
RF : q=1	0.306	0.308
RF : q=4	0.242	0.284
RF : q=8	0.238	0.276
RF : q=16	0.266	0.286
Validation croisée : 10 répétitions	0.234	0.272

À la lecture du tableau, on peut voir que l'erreur OOB (Out Of Bag) est la plus basse avec notre modèle tuné. Les paramètres de notre modèle tuné sont 500 arbres et `mtry`= 16. Toutefois, l'erreur OOB pour `mtry`= 8 et un nombre d'arbres égal à 200 est quasiment similaire. On choisi donc le modèle avec 200 arbres et 8 mtry qui est moins coûteux pour une erreur quasiment égale.



On remarque que l'erreur pour la classe des bonnes moyennes est beaucoup moins importante que l'erreur pour la classe des mauvaises moyennes. Ce qui nous donne une erreur OOB général autour de de 0.28.

4.2.1 Importance des variables



On remarque que l'on a toujours les mêmes variables qui jouent un rôle important : les variables `failures`, `matiere`, `higher`, `Walc`.

4.3 Bagging

Pour le bagging, c'est la même méthode que pour la forêt aléatoire mais on utilise `mtry`= nvar, notre nombre de variable totale.

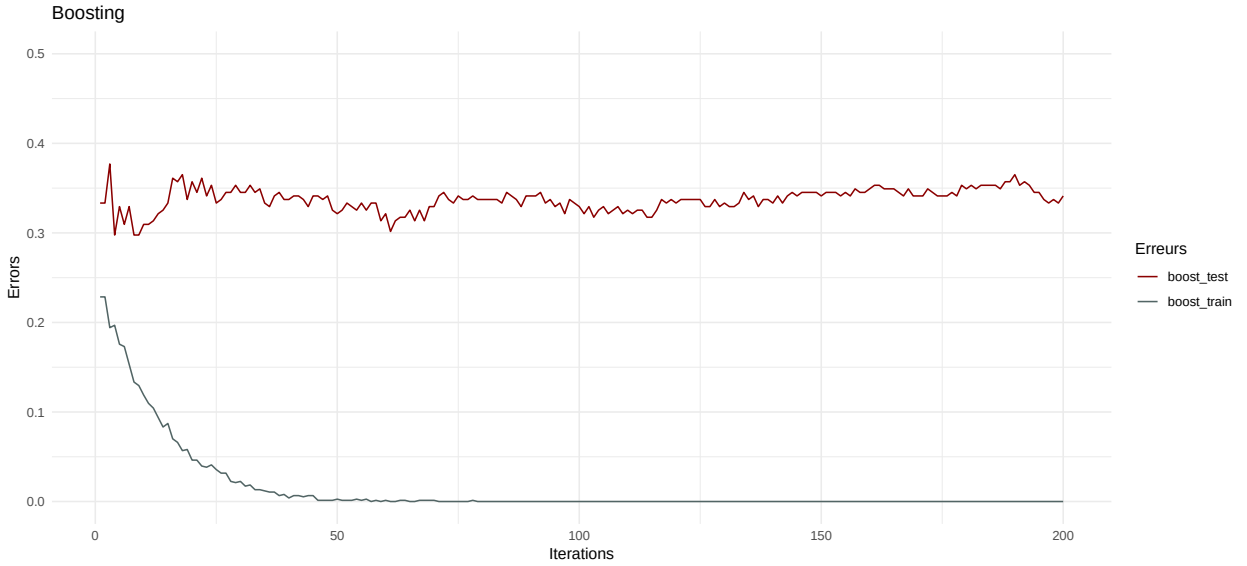
Table 16 – Erreur bagging en fonction du nombre d'arbres

	erreur moyenne_OOB
500 arbres	0.309
400 arbres	0.304
300 arbres	0.318
200 arbres	0.324
100 arbres	0.328

Notre meilleur modèle de bagging semble donc être celui avec 400 arbres. Nous allons donc créer ce modèle et le tester sur nos données test afin d'avoir notre erreur. Finalement, avec notre bagging, nous avons une erreur sur les données test de 0.286.

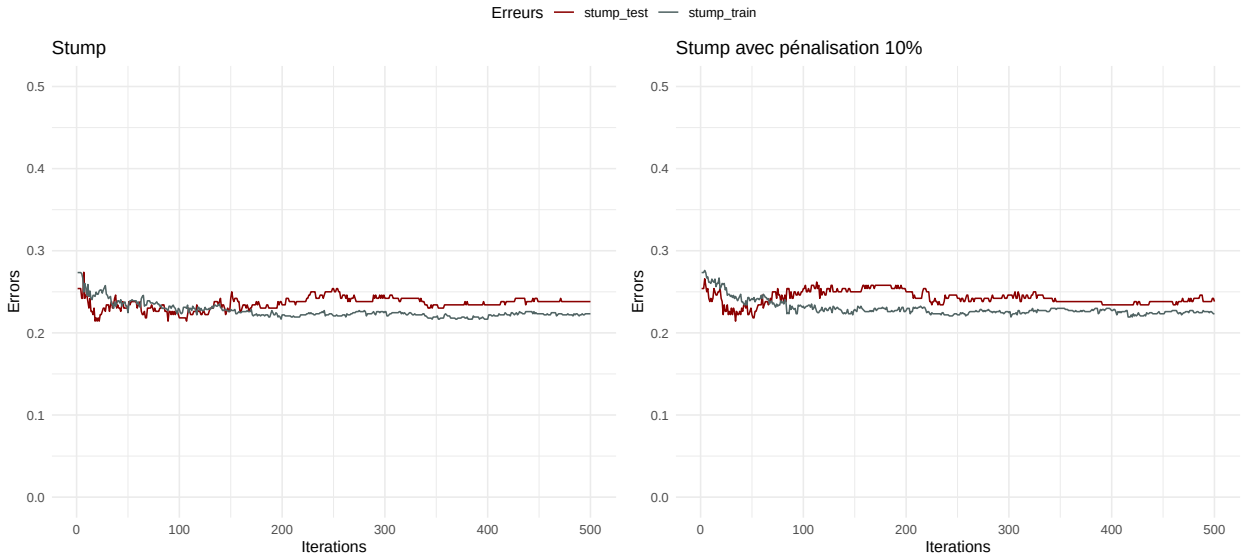
4.4 Boosting

Pour le boosting, nous allons chercher le meiller modèle parmi : le boosting sur arbre complet, le boosting avec stumps non pénalisés et pénalisés ainsi que boosting simple avec pénalisation. Les pénalisations seront de 0.1 et 0.01.



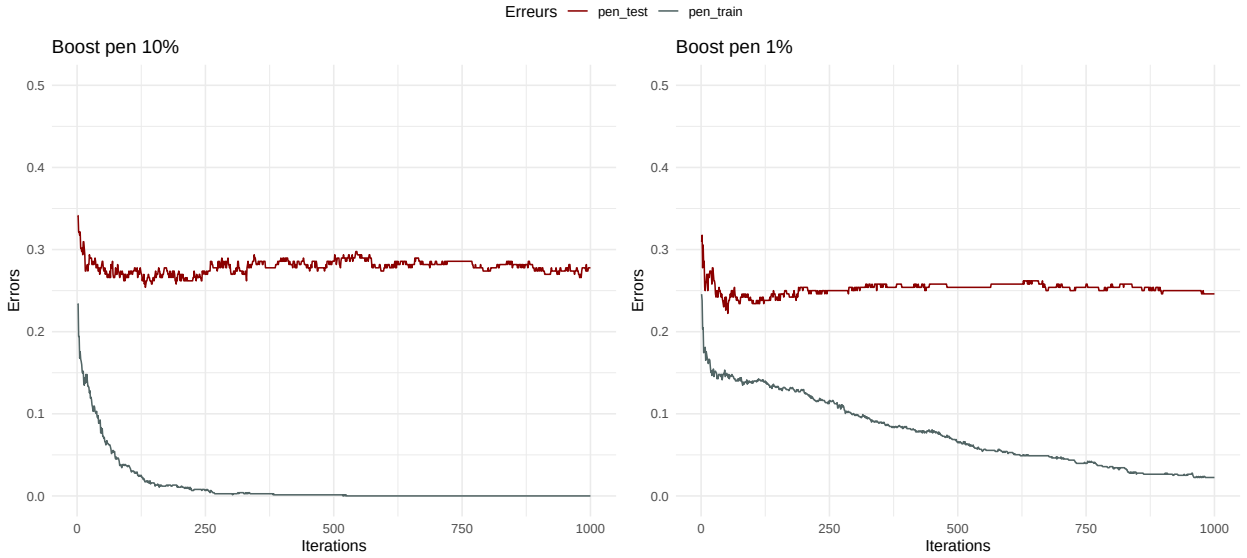
Ici, l'erreur d'entrainement descend très vite à zéro alors que l'erreur en test reste autour de 0.3-0.35. Ce modèle n'est donc pas optimal.

4.4.1 Stumps

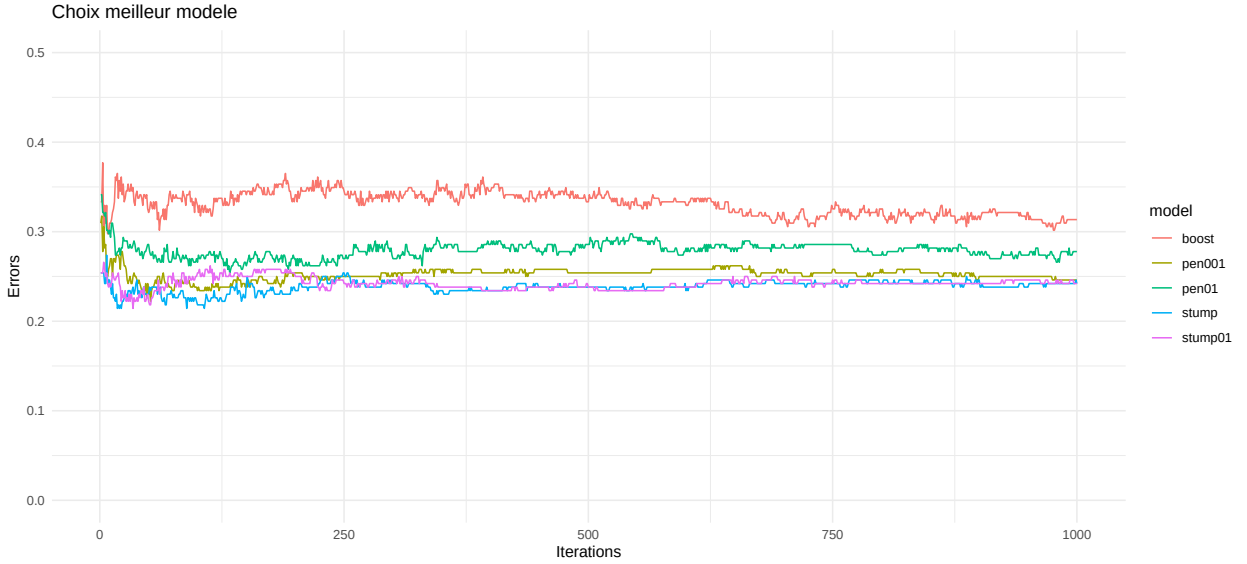


Comparé au boosting simple, la différence entre l'erreur en test et l'erreur d'entrainement est très faible pour les stumps. Les deux types d'erreur se situe entre 0,2 et 0,3. Si nous rajoutons une pénalisation de 10 %, les oscillations d'erreurs sont un peu plus importante et cela jusqu'à 200 itérations.

4.4.2 Boosting avec pénalisation



En pénalisant le modèle de boosting simple, on voit que les deux courbes des différentes erreurs sont plus proches avec une pénalisation plus faible. Ceci est normal car en pénalisant, on apprend moins vite. On peut également constater que l'erreur est plus basse pour une pénalisation à 1 % qu'un boosting simple ou un boosting avec une pénalisation à 10 %.



Grâce à ce graphique, on remarque que le modèle boosting sur arbre complet a une erreur qui est plus élevée que les autres. Quand on applique la pénalisation, l'erreur diminue un peu mais elle est toujours plus élevée qu'un boosting avec des stumps. Les deux modèles avec stumps, que ce soit avec ou sans pénalisation, sont quasiment identiques.

Faisons un tableau récapitulatif des erreurs pour un nombre d'itérations défini. Ceci va nous permettre de determiner quel est le meilleur modèle.

TABLE 17 – Choix du meilleur boosting en fonction de l'erreur en test et du nombre d'iterations

	50	100	200	500	1000
Boosting	0.321	0.329	0.341	0.333	0.313
Stump	0.230	0.218	0.234	0.238	0.242
Stump pen10%	0.222	0.254	0.250	0.238	0.242
Boost Pen 10%	0.286	0.266	0.270	0.286	0.278
Boost Pen 1%	0.230	0.234	0.254	0.254	0.246

Avec l'aide de ce tableau récapitulatif et du graphique que nous avons vu auparavant, les deux modèles avec stumps ont une erreur en test plus basse que pour tous les autres modèles à un nombre d'itération donnée. On constate que le meilleur boosting est donc : **Boosting avec stumps et 100 itérations soit une erreur en test de** 0.218.

4.5 Comparaison des meilleurs modèles

Danc cette dernière sous-partie, nous allons comparer tous nos meilleurs modèles de chaque famille pour determiner notre modèle final.

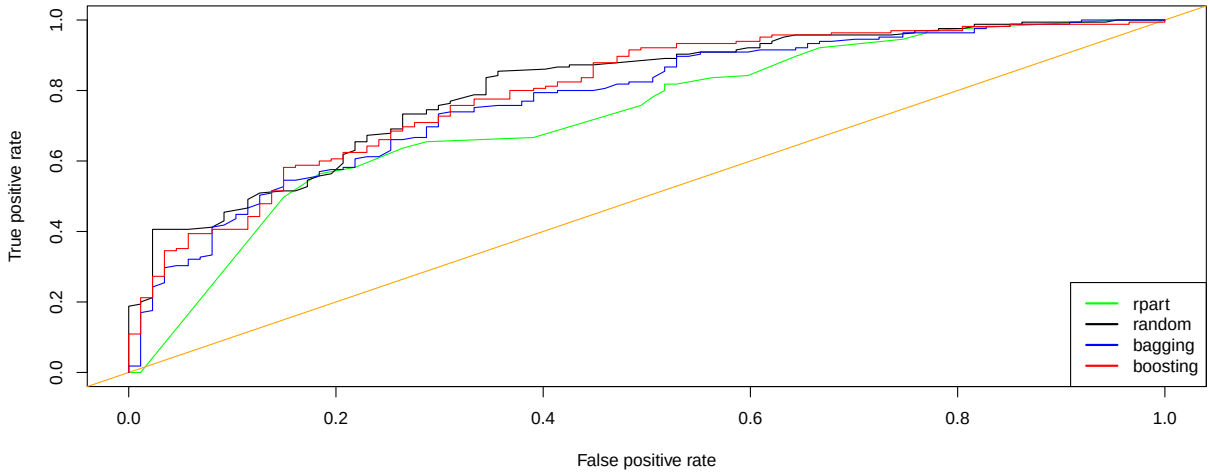


TABLE 18 – Résumé final et choix du modèle

	Erreurs globales	Erreurs < 10	Erreur > 10	AUC
Arbre de décision	0.298	0.517	0.182	0.727
Random forest	0.254	0.575	0.091	0.806
Bagging	0.282	0.506	0.164	0.775
Boosting	0.234	0.448	0.121	0.797

Le meilleur modèle entre toutes les familles de la classification non supervisée est donc le modèle boosting. Il prédit le mieux sur les moyennes basses et c'est celui qui a l'erreur globale la plus faible. Pour les moyennes hautes, c'est le modèle de random forest qui prédit le mieux mais également celui qui prédit le moins bien les moyennes basses. Le modèle retenu sera donc boosting avec stumps et 50 itérations.

5 Entrainement sur notre base complète de départ

Dans cette partie, nous utilisons nos meilleurs modèles (LDA et boosting) sur notre base de données de départ afin d'entrainer au maximum nos modèles et que ceux-ci soient prêts à l'usage afin de prédire les notes de futurs étudiants selon leurs caractéristiques.

6 Conclusion

Au final, nos meilleurs modèles pour prédire la moyenne des étudiants sont le boosting et la LDA. Après entrainement de nos deux modèles sur toutes nos données, nous pensons qu'il est mieux de garder le boosting comme modèle à enregistrer pour prédire si dans le futur de nouveaux étudiants veulent savoir leur moyenne en fonction de leurs caractéristiques. En effet, le modèle LDA demande de faire une ACM en amont et utilise seulement les coordonnées des individus. Si dans le futur, nous voulons créer une interface où l'étudiant rentre ses caractéristiques, il ne pourra pas rentrer de coordonnées.

7 Annexe :

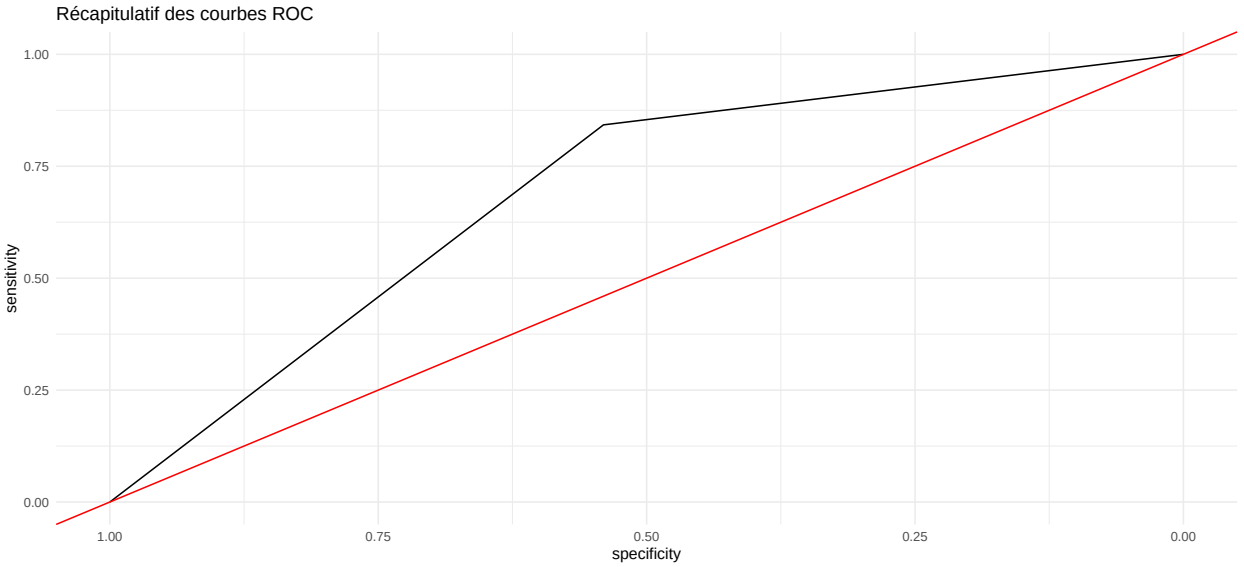
7.1 MDA

Table 19 – Matrice de confusion MDA

	< 10	> 10
< 10	47	40
> 10	26	139

Table 20 – Matrice de confusion (proportions)

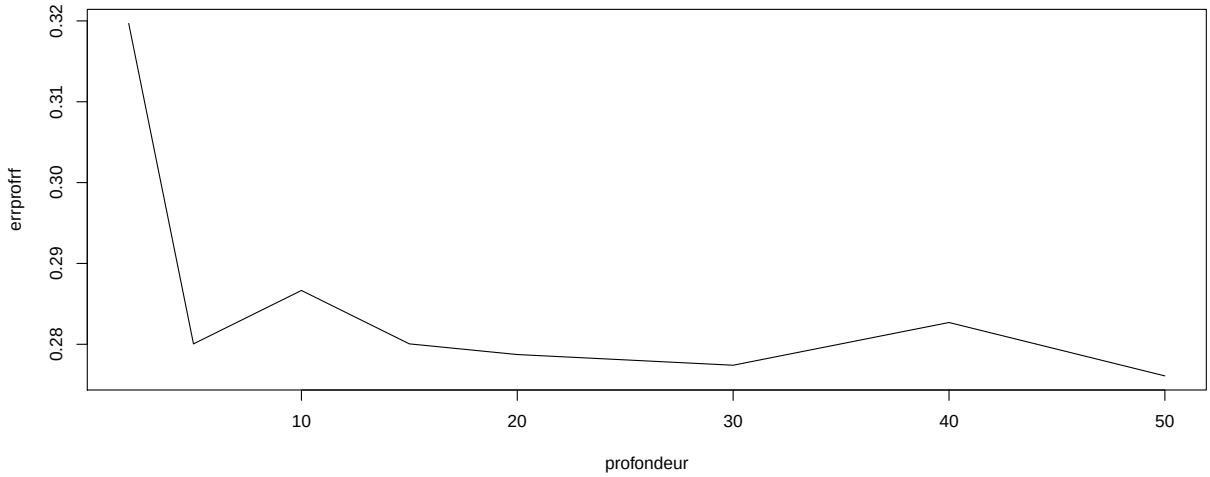
	< 10	> 10
< 10	0.5402299	0.4597701
> 10	0.1575758	0.8424242



Pour la LDA, on suppose que les classes suivent une distribution normale (Gaussienne), pour la MDA, on va supposer que chaque classe est un mélange gaussien de sous-classes, d'où le nom "mixture discriminante analysis". Toutefois, cette méthode n'a pas été vu en cours et ne peut pas être expliqué très clairement. Nous avons décidé de juste mettre ce modèle en annexe afin de voir si il performe mieux ou pas que LDA et QDA. Le modèle nous donne une erreur globale de 0.262. Il ne semble donc pas performer de manière plus efficace que LDA et QDA.

7.2 Profondeur arbre random forest

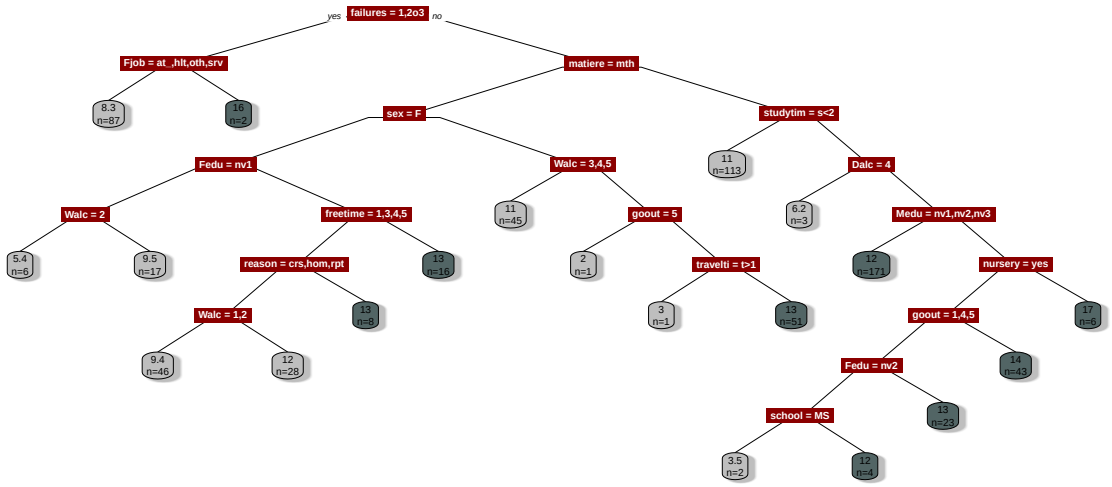
Voici le graphique pour nous aider dans la recherche de la profondeur de l'abre en random forest, dont les résultats étaient très aléatoires.



7.3 Arbre de régression

7.3.1 Un premier arbre

Voici un test d'abre de régression, c'est-à-dire que l'on prédit une variable quantitative (ici notre variable `moyenne`).



7.3.2 Random forest

Le meilleur modèle de random forest pour une regression est donc : 500 arbres et 16 mtry. On a une erreur MSE de 7.45